Chapitre 3
Systèmes d'équations algébriques

Mathématiques-Informatique, Sciences de l'Homme et de la Société

Université Moulay Ismail - Meknès

11 décembre 2011

On a résolu une équation de la forme

$$f(x) = 0 \quad \text{où } x \in \mathbb{R} \text{ et } f(x) \in \mathbb{R} \quad \text{(i.e. } f : \mathbb{R} \to: \mathbb{R})$$

On complique encore les choses : deux grandes familles de problèmes possibles :

- les systèmes d'équations linéaires de n équations et n inconnues :

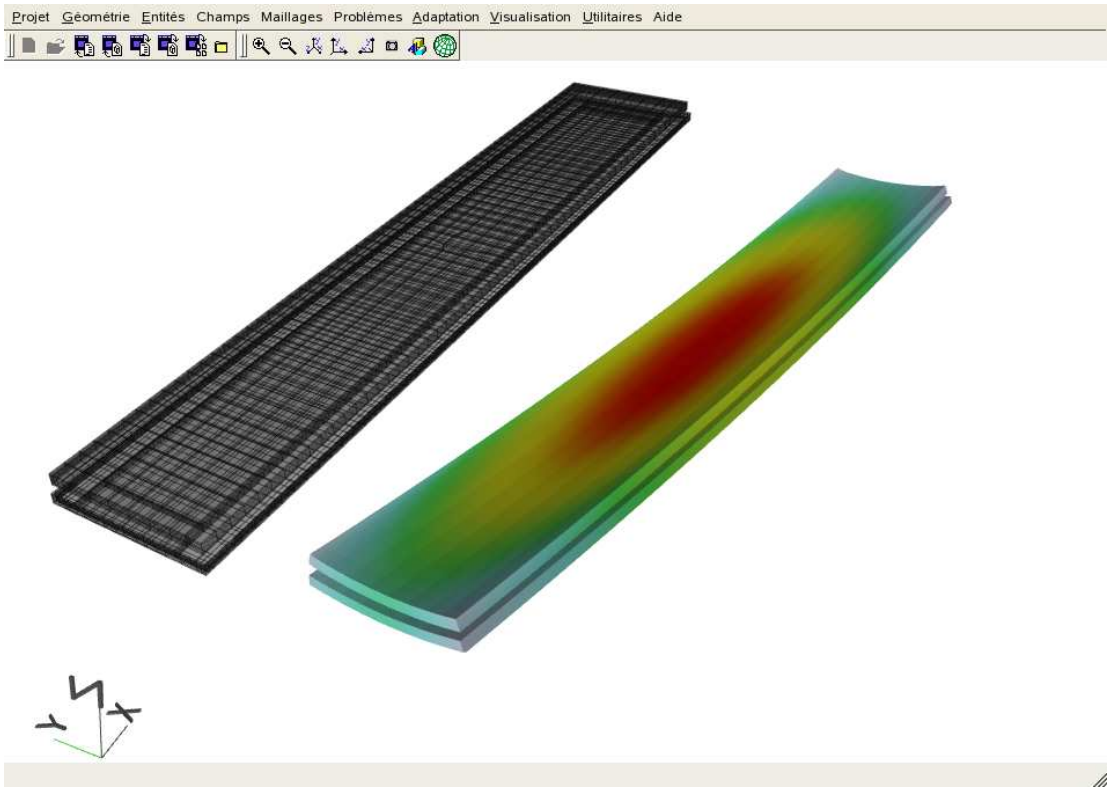
  $$Ax = b$$

  où A est une matrice carrée inversible et b un vecteur
  ($F : \mathbb{R}^n \to: \mathbb{R}^n$ avec $F(x) = Ax - b$ et on cherche $F(x) = 0$)
- les systèmes non linéaires : $F(x) = 0$ où $F : \mathbb{R}^n \to: \mathbb{R}^n$ est une application à n variables : n équations non-linéaires dépendant de n variables.

Beaucoup d'applications dans le domaine de l'ingénierie exigent de résoudre des systèmes d'équations qui comportent souvent un grand nombre d'inconnues. On rencontre ces applications, par exemple, dans les domaines suivants :

- la mécanique des fluides : écoulement de l'air autour d'un avion, écoulement dans une turbine hydraulique, diffusion dans les artères d'un médicament enrobant des stents, etc.
- la mécanique des solides : déformation d'un matériau soumis à des contraintes extérieures, analyse des structures complexes, etc.
- Optimisation de la distribution des fréquences sur des antennes, filtrage de signaux, etc.

Comportement hygro-mécanique de lames de parquet en services (période de 42 jours, humidité relative augmentée de 20%).



Ce calcul exige de résoudre un premier système linéaire de 35890 équations pour le mouvement de l'humidité puis un système de 107670 équations pour la déformation. Ces deux résolutions sont faites au total 129 fois pour décrire le comportement à divers moment dans le temps.

Un système d'équations linéaires est un système de n équations linéaires à n inconnues

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n &= b_2 \\ \vdots & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n &= b_n \end{cases}$$

où a_{ij} et b_j sont connus pour $i, j = 1, ..., n$.

Exemple : Trouver x_1, x_2, x_3 solution de

$$\begin{cases} 2x_1 + 3x_2 - x_3 = 5 \\ 4x_1 + 4x_2 - 3x_3 = 3 \\ -2x_1 + 3x_2 - x_3 = 1 \end{cases}$$

Écriture matricielle :

$$Ax = b$$

où A dénote la matrice carrée d'ordre n des coefficients du système linéaire et b le vecteur du second membre.

$$A = \begin{pmatrix} a_{11} & \dots & a_{1i} & \dots & a_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & \dots & a_{ii} & \dots & a_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{ni} & \dots & a_{nn} \end{pmatrix} \qquad b = \begin{pmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_n \end{pmatrix} \qquad x = \begin{pmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{pmatrix}$$

Exemple : Trouver $x \in \mathbb{R}^3$ solution de

$$\begin{pmatrix} 2 & 3 & -1 \\ 4 & 4 & -3 \\ -2 & 3 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \\ 1 \end{pmatrix}$$

Rappel d'algèbre :

Trois énoncés équivalents :

- Le système $Ax = b$ admet une solution unique si et seulement si la matrice A est de rang maximal (autant d'inconnues que d'équations et pas d'équation qui se répète).
- Le système $Ax = b$ admet une solution unique si et seulement si $\det A \neq 0$
- Le système $Ax = b$ admet une solution unique si et seulement si A est inversible.

Dans ce cas la solution est fournie par

- $x = A^{-1}b$
- La formule de Cramer : $x_i = \dfrac{\det A_i}{\det A}$
  où A_i est la matrice A dont la colonne i est remplacée par b.

Remarque

- La plupart du temps on traite des cas où la matrice est inversible (matrice non-singulière). Ce qui assure l'existence d'une solution.
- Sauf dans de rare cas la formule de Cramer peut être considérer comme purement théorique (sans applications).
- Sauf dans des cas particuliers il est plus rapide et simple de résoudre le système linéaire que d'inverser la matrice (même "à la main").

Comment résoudre efficacement des systèmes linéaires ?

Méthode des substitutions successives

La méthode la plus simpliste (et classique) consiste a éliminer des inconnues dans les équations par substitution successive :

$$\begin{pmatrix} 2 & 3 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 8 \\ 11 \end{pmatrix}$$

De la première équation on a

$$x_1 = \frac{8 - 3x_2}{2}$$

Et en substituant dans la deuxième (éliminant x_1) on a

$$3(\frac{8 - 3x_2}{2}) + 4x_2 = 11 \Rightarrow x_2 = 2$$

Pour une matrice quelconque cette approche est simple pour un humain mais difficile à traduire en algorithme. On doit abandonner cette approche du moins sous cette forme.

Cette approche est plus simple à décrire pour certaines structures de matrices.

Matrice diagonale

Facile lorsque le système linéaire a une matrice diagonale :

$$Ax = \begin{pmatrix} a_{11} & 0 & 0 & \dots & 0 \\ 0 & a_{22} & 0 & \dots & 0 \\ \vdots & & & \ddots & 0 \\ 0 & \dots & 0 & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} \Leftrightarrow a_{ii}x_i = b_i \quad i = 1, \dots, n$$

alors $x_i = \dfrac{b_i}{a_{ii}}$. Solution unique ssi $a_{ii} \neq 0 \ \forall i$ $(\det A = \prod_{i=1}^{n} a_{ii} \neq 0)$

Définition 3.1

Une matrice est dite **triangulaire inférieure (ou supérieure)** si toutes ces entrées a_{ij} **sont nulles pour** $i < j$ **(**$i > j$ **resp.)**.

Triangulaire inférieure = “nulle au dessus de la diagonale“.
Triangulaire supérieure = “nulle au dessous de la diagonale“.

$$\begin{pmatrix} a_{11} & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & 0 & \dots & 0 \\ a_{31} & a_{32} & a_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} \qquad \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ 0 & 0 & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{pmatrix}$$

Matrice triangulaire

Les matrices triangulaires : résoudre sucessivement en choisissant l'équation avec le moins d'inconnues.

- Matrice triangulaire inférieure : **descente triangulaire** (on commence à 1 on fini à n "en bas").

$$x_1 = \frac{b_1}{a_{11}} \qquad x_i = \frac{b_i - \sum_{k=1}^{i-1} a_{ik} x_k}{a_{ii}} \quad i = 2, ..., n$$

- Matrice triangulaire supérieure : **remontée triangulaire** (on commence à n on fini à 1 "en haut").

$$x_n = \frac{b_n}{a_{nn}} \qquad x_i = \frac{b_i - \sum_{k=i+1}^{n} a_{ik} x_k}{a_{ii}} \quad i = n-1, n-2, ..., 1$$

Unicité dans le cas des matrices triangulaires

La remontée et la descente triangulaire n'ont de sens qu'a la condition que les termes diagonaux soient tous non nuls.

Que se passe-t'il si cela n'est pas le cas ?

Puisque le déterminant d'une matrice triangulaire est le produit des termes de la diagonale, alors

$$\det A = \prod_{i=1}^{n} a_{ii} \neq 0 \Leftrightarrow a_{ii} \neq 0 \quad \forall i$$

Donc l'unicité de la solution impose que les termes diagonaux soient tous non nuls. Dans le cas contraire le système n'a pas une solution unique et A n'est pas inversible.

Exemple de remontée triangulaire

Soit à résoudre le système

$$\begin{pmatrix} 3 & 1 & 3 \\ 0 & 2 & 2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 30 \\ 15 \end{pmatrix}$$

$$x_3 = \frac{b_3}{a_{33}} = 15 \quad x_2 = \frac{b_2 - \sum_{k=3}^{3} a_{2k} x_k}{a_{22}} = \frac{30 - 2x_3}{2} = 0$$

$$x_1 = \frac{b_1 - \sum_{k=2}^{3} a_{1k} x_k}{a_{11}} = \frac{0 - 1x_2 - 3x_3}{3} = -x_3 = -15$$

L'algorithme de remontée (ou de descente) est efficace. On sait que l'on peut transformer un système quelconque en système triangulaire.

On voudrait

- une méthode de passage d'un système de quelconque vers un système triangulaire tout en conservant la solution du système original.
- utiliser la remontée ou descente triangulaire.

Pour cela on va revoir les opérations élémentaires sur les systèmes linéaires. On produira ensuite deux **méthodes directes** : la méthode de Gauss et la méthode de la décomposition LU.

Definition 3.2

Une méthode de résolution d'un système linéaire est dite **directe** si la solution du système peut être obtenue en **un nombre fini et prédéterminé d'opérations**.

Remarque sur les méthodes directes

- Une méthode directes est essentiellement une méthode basée sur la remontée (descente). Pour de telle méthode on peut faire un estimé a priori du nombre d'opérations en virgule flottante et du temps de calcul (une remontée ou une descente exige $\approx n^2$ opérations élémentaires). La plupart du temps la limitation dans l'utilisation de méthodes directes vient de l'espace mémoire requis pour la résolution.
- À l'opposée de ces méthodes on trouve les méthodes itératives. Ces méthodes ne garantissent pas la convergence et ne peuvent garantir un nombre fixe d'opérations pour l'obtention d'une solution. Cependant la plupart des méthodes itératives exigent moins d'espace mémoire.

Transformation = produit par une matrice

- On veut garder la linéarité du système alors on transforme avec les opérations de bases $(+, -, \times, \div)$ avec des scalaires sur et entre les lignes du système.
- Une transformation de ce type est représentable par une matrice multipliant la matrice A et le vecteur b du système :

$$WAx = Wb$$

- Quels conditions sur W ? Si on a la solution (unique) de $WAx = Wb$ alors on veut que cette solution soit la solution de $Ax = b$. Alors il faut que la matrice W soit inversible !
- On s'assurera de la validité de notre transformation en s'assurant que son déterminant est non nul (ou quel est inversible).

Exemple

Soit

$$Ax = \begin{pmatrix} 10 & 6 & 2 \\ 4 & 4 & 1 \\ 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 44 \\ 21 \\ 14 \end{pmatrix}$$

Prenons

$$W = \begin{pmatrix} 1 & -1 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \Rightarrow WA = \begin{pmatrix} 3 & 0 & 0 \\ 1 & 2 & 0 \\ 3 & 2 & 1 \end{pmatrix} \quad Wb = \begin{pmatrix} 9 \\ 7 \\ 14 \end{pmatrix}$$

la solution $x = (3\,2\,1)^T$ correspond à la solution du système original.

$$\tilde{W} = \begin{pmatrix} 1 & -1 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix} \Rightarrow \tilde{W}A = \begin{pmatrix} 3 & 0 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \tilde{W}b = \begin{pmatrix} 9 \\ 7 \\ 0 \end{pmatrix}$$

Le système n'est plus valide : pas unicité donc pas correspondance entre les deux problèmes !

Pour réduire un système linéaire à la forme triangulaire on utilise 3 opérations élémentaires sur les lignes

- Multiplication d'une ligne L_i par une scalaire λ non nul :

$$\lambda L_i \longrightarrow L_i$$

- Addition de deux lignes :

$$L_i + \lambda L_j \longrightarrow L_i$$

- permuter deux lignes :

$$L_i \longleftrightarrow L_j$$

On sait qu'en multipliant (à gauche) le système original par une matrice inversible la solution sera inchangée. On va montrer que les 3 opérations proposées correspondent a des produits par une matrice inversible, nous assurant ainsi de conserver la solution originale.

A l'opération multiplication de la ligne i par λ non nul : $\lambda L_i \longrightarrow L_i$ correspond le produit de $Ax = b$ avec la matrice élémentaire M :

$$m_{jj} = 1 \;\; j = 1, ..., i-1, 1+1, ..., n \quad m_{ii} = \lambda \quad m_{jk} = 0 \text{ sinon}$$

On sait que $\det M = \lambda$ donc la matrice est inversible si $\lambda \neq 0$

$$M = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ 0 & \ddots & 0 & \dots & 0 \\ 0 & \dots & \lambda & \dots & 0 \\ 0 & & \dots & 1 & \dots \\ 0 & \dots & & \dots & 1 \end{pmatrix} \qquad M^{-1} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ 0 & \ddots & 0 & \dots & 0 \\ 0 & \dots & \frac{1}{\lambda} & \dots & 0 \\ 0 & & \dots & 1 & \dots \\ 0 & \dots & & \dots & 1 \end{pmatrix} \leftarrow i$$

A l'opération intervertir deux lignes du système : $L_i \longleftrightarrow L_j$ correspond le produit de $Ax = b$ avec la matrice élémentaire P (matrice de permutation) obtenue en permutant les lignes i et j de la matrice identité

$$P = \begin{matrix} \\ j \rightarrow \\ i \rightarrow \\ \\ \end{matrix} \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ 0 & \ddots & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & 0 \\ 0 & & 1 & 0 & \dots \\ 0 & \dots & & \dots & 1 \end{pmatrix}$$

Puisque $PP = I$ la matrice P est inversible et son inverse $P^{-1} = P$. On a aussi que $\det P = -1$

A l'opération addition de deux lignes $L_i + \lambda L_j \longrightarrow L_i$ correspond le produit de $Ax = b$ avec la matrice élémentaire T

$$t_{kk} = 1 \;\; k = 1, ..., n \quad t_{ij} = \lambda \quad t_{kl} = 0 \text{ sinon}$$

On sait que $\det T = 1$ alors T est inversible (il suffit d'enlever ce que l'on a ajouter)

$$T = \begin{matrix} \\ j \rightarrow \\ i \rightarrow \\ \\ \\ \end{matrix} \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ 0 & \ddots & 0 & \dots & 0 \\ 0 & \dots & 1 & \dots & 0 \\ 0 & & \lambda & 1 & \dots \\ 0 & \dots & & \dots & 1 \end{pmatrix} \qquad T^{-1} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ 0 & \ddots & 0 & \dots & 0 \\ 0 & \dots & 1 & \dots & 0 \\ 0 & & -\lambda & 1 & \dots \\ 0 & \dots & & \dots & 1 \end{pmatrix}$$

La méthode d'élimination de Gauss consiste à utiliser les opérations élémentaires afin de réduire le système sous la forme triangulaire supérieure.

On distingue deux cas :

- Sans pivotement : on ne permet pas la permutation de deux lignes (la transformation P est interdite) ; dans certain cas on ne pourra pas obtenir de matrice triangulaire sup. !
- Avec pivotement : on permet la permutation des lignes ; dans ce cas on obtiendra toujours une matrice triangulaire sup. si la matrice est inversible.

Dans la pratique on ne construit pas les matrices (de types M,T et P) nécessaires à la transformation.

Cependant la méthode de Gauss est valide parce qu'elle consiste à multiplier le système de départ par une suite de matrices inversibles. On verra plus tard comment ces matrices nous seront utiles.

Dans l'application de la méthode de Gauss on procède par élimination des termes sous la diagonale en utilisant la notion de matrice augmentée.

Définition

La matrice augmentée du système $Ax = b$ est la matrice de dimension $n \times n + 1$ obtenu en ajoutant le membre de droite b à la matrice A :

$$\left[\begin{array}{ccccc|c} a_{11} & \dots & a_{1i} & \dots & a_{1n} & b_1 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ a_{i1} & \dots & a_{ii} & \dots & a_{in} & b_i \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ a_{n1} & \dots & a_{ni} & \dots & a_{nn} & b_n \end{array}\right]$$

Cette notation est très utile puisque les opérations élémentaires doivent être faites sur la matrice A et sur b simultanément.

- Un système $Ax = b$ a une solution unique ssi
  - det$A \neq 0$
  - A est inversible (non singulière)
  - # équations = # inconnues et pas déquations répétées
- Les systèmes diagonaux et trinagulaires (inf. et sup.) peuvent être résolus par “remontée” ou “descente” triangulaire.
- Transformation d’un système quelconque en système triangulaire : on veut passer de $Ax = b$ vers

$$\tilde{A}x = \tilde{b}$$

avec

- $\tilde{A}$ triangulaire
- la solution des deux système doit être la même

$$Ax^* = b \Leftrightarrow \tilde{A}x^* = \tilde{b}$$

- On a montrer que les bonnes transformations sont des produits par des matrices inversibles et que les 3 opérations de bases :
  - multiplication d'une ligne par un scalaire mène à une matrice M inversible
  - addition de deux lignes mène à une matrice T inversible
  - permutation de deux lignes mène à une matrice P inversible

  Avec ces trois opérations on peut transformer un système en système triangulaire.
- Méthode de Gauss : utilisation de combinaisons de M, T et P pour obtenir une matrice triangulaire supérieure. Deux choix :
  - Pas de permutations permises : peux mener à des matrices non-triangulaires
  - avec permutation : on peut toujours obtenir une matrice traingulaire.

  Dans la pratique cette méthode ne tient pas compte des matrices de transformation. Elles ne sont là que pour démontrer que la méthode de Gauss transforme de la bonne manière le système de départ.

Exemple 3.8

On applique la méthode de Gauss sur la matrice augmentée

$$A = \left[\begin{array}{ccc|c} 2 & 1 & 2 & 10 \\ 6 & 4 & 0 & 26 \\ 8 & 5 & 1 & 35 \end{array}\right]$$

Avec une permutation : Exemple 3.9

On applique la méthode de Gauss sur la matrice augmentée

$$A = \left[\begin{array}{cccc|c} 1 & 1 & 2 & 1 & 2 \\ 2 & 2 & 5 & 3 & 4 \\ 1 & 3 & 3 & 3 & -2 \\ 1 & 1 & 4 & 5 & -2 \end{array}\right]$$

Et si je veux changer b ?

Trois options :

1. Je recommence le processus avec la nouvelle valeur de b car je dois transformer A et b simultanément dans la méthode de Gauss
2. j'attend d'avoir TOUT les b que je veux et je fais Gauss sur la matrice augmentée de tout les b
3. J'essaie de garder l'information ayant servi à la transsformation de la matrice lors de la première rsolution pour résoudre avec les autres b

Revenons à nos exemples pour voir ce qu'on peut faire en tenant compte des matrices T ,M et P cette fois

Exemple 3.8 revisité

On applique la méthode de Gauss sur la matrice augmentée

$$A = \left[\begin{array}{ccc|c} 2 & 1 & 2 & 10 \\ 6 & 4 & 0 & 26 \\ 8 & 5 & 1 & 35 \end{array}\right]$$

Mais on construit les 3 matrices T et on a

$$A = (T_1^{-1} T_2^{-1} T_3^{-1}) U = LU$$

Avec une permutation : Exemple 3.9 revisité

On applique la méthode de Gauss sur la matrice augmentée

$$A = \left[\begin{array}{cccc|c} 1 & 1 & 2 & 1 & 2 \\ 2 & 2 & 5 & 3 & 4 \\ 1 & 3 & 3 & 3 & -2 \\ 1 & 1 & 4 & 5 & -2 \end{array}\right]$$

Dans ce cas on a des T et un P et on a

$$PA = (P_4 T_1^{-1} T_2^{-1} T_3^{-1} P_4 T_5^{-1})U = LU \Leftrightarrow A = PLU$$

La décomposition LU permet de résoudre avec différents b

Cette décomposition donne toute l'information nécessaire pour résoudre à "répétition". Puisque L et U sont construit on a

$$Ax = b \Leftrightarrow (LU)x = b \Leftrightarrow L(Ux) = b$$

Si on pose $y = Ux$ alors la résolution de $Ax = b$ peut se faire en deux temps :

1. résoudre $Ly = b$ par une descente triangulaire
2. résoudre $Ux = y$ par une remontée triangulaire

Une fois le problème résolu pour un b, en ayant garder L je peux changer le b et résoudre à nouveau sans refaire de transformation du système.

Remarques

- la méthode de Gauss est une décomposition LU où on “jete“ L. Plus la peine d’en parler (sauf pour les exercices :-)
- les permutations causent des problèmes (empêchent d’avoir un LU ”propre”) on en reparlera.
- Pour le moment on va faire comme si A était permuter avant qu’on arrive...

Unicité de LU

La décomposition LU se fait grâce aux transformations de $Ax = b$ mais on pourrait ajouter des transformations (superflues). Par exemple on pourrait finir en s’assurant que les pivots (termes diagonaux non nuls) soit tous égaux à 1 en appliquant des transformations M. Donc la décomposition LU n’est pas unique.

Pour éviter les ambiguités possible (l'opérateur == est mal défini) on voudrait assurer l'unicité de la décomposition. Pour se faire on va imposer une conditions supplémentaire à notre décomposition.
Deux méthodes "classiques"

- Décomposition de type Matlab : $L_{ii} = 1 \quad \forall i$ (pas de division dans la descente)
- Décomposition de Crout : $U_{ii} = 1 \quad \forall i$ (pas de division dans la remontée)

Décomposition de Crout

Algorithme composé de trois parties

1. factorisation LU avec $U_{ii} = 1 \quad \forall i$
2. Résolution de $Ly = b$ par descente triangulaire
3. Résolution de $Ux = y$ par remontée triangulaire

Comment on fait la factorisation

On identifie termes à termes les composantes de L et U. À noter que c'est un système linéaire de n^2 équations et inconnues qui a une solution unique quand A est inversibles.
Une particularité de l'approche est que l'on résout alternativement pour une colonne de L puis pour une ligne de U en commençant par la colonne 1 donc on construit :

$$L_{i1}, U_{1i}, ..., L_{in-1}, U_{n-1i}, L_{in}$$

Cette stratégie nous permettra de réduire l'espace mémoire lors de la factorisation.

- Factorisation
  - $L_{i1} = A_{i1} \quad i = 1, ..., n,\ U_{1i} = \frac{A_{1i}}{L11} \quad i = 2, ..., n$
  - $i = 2, ..., n-1$
    - $L_{ii} = A_{ii} - \sum_{k=1}^{i-1} L_{ik} U_{ki}$
    - $j = i+1, ..., n$

$$L_{ji} = A_{ji} - \sum_{k=1}^{i-1} L_{jk} U_{ki}$$

$$U_{ij} = \frac{A_{ij} - \sum_{k=1}^{i-1} L_{ik} U_{kj}}{L_{ii}}$$

  - $L_{nn} = A_{nn} - \sum_{k=1}^{n-1} L_{nk} U_{kn}$
- descente sur $Ly = b$
  - $y_1 = \frac{b_1}{L_{11}} \quad y_i = \frac{b_i - \sum_{k=1}^{i-1} L_{ik} y_k}{L_{ii}} \quad i = 2, ..., n$
- remonté sur $Ux = y$
  - $x_n = y_n \quad x_i = y_i - \sum_{k=i+1}^{n} U_{ik} x_k \quad i = n-1, ..., 1$

Remarque

L'algo ne fonctionne que si $L_{ii} \neq 0 \quad \forall i$.

- Si on a pas fait de permutations alors $\det A = \det L \det U$ mais $\det U = 1$ et

$$\det A \neq 0 \Leftrightarrow \det L \neq 0 \Leftrightarrow L_{ii} \neq 0 \quad \forall i$$

- Si on a fait des permutations $\det A = \det P \det L \det U$ Puisque $\det P \neq 0$ alors

$$\det A \neq 0 \Leftrightarrow \det L \neq 0 \Leftrightarrow L_{ii} \neq 0 \quad \forall i$$

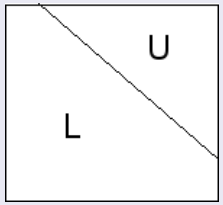
Donc si A est inversible l'algorithme va bien.

Remarque

D'un point de vue pratique une fois la factorisation LU faite on a plus de raison de conserver la matrice A. En fait notre procédure de factorisation est faite pour que l'on puisse "écraser" A au fur et à mesure de la factorisation. Ce qui permet de minimiser l'espace de stockage puisqu'on a pas simultanément A, L et U

Definition 3.4

La **notation compacte de LU** consiste a remplacer la matrice A par la matrice



Les coefficients de la diagonale de U ne sont pas gardé puisqu'ils sont tous égaux à 1 mais on ne les néglige pas pour autant.

Cette notation revient à stocker :

$$L + (U - I) = L + U - I$$

à la place de A

Exemple 3.11

Utilisation de la décomposition LU avec notation compacte pour

$$A = \begin{bmatrix} 3 & -1 & 2 \\ 1 & 2 & 3 \\ 2 & -2 & -1 \end{bmatrix} \quad b = \begin{bmatrix} 12 \\ 11 \\ 2 \end{bmatrix}$$

Permutations

On gardera un vecteur de permutation (position ou numérotation) qui indiquera les permutations effectuées sur la matrice. A la fin de la factorisation on obtient L, U et le vecteur de permutation.

Si nécessaire on reconstruit la matrice P en appliquant les permutations sur une matrice identié. Mais c'est normalement inutile.

Comment résoudre une fois la factorisation terminée ?

$$Ax = b \text{ et } PA = LU \Leftrightarrow PAx = LUx = Pb \Leftrightarrow Ly = Pb \text{ et } Ux = y$$

On ne construit pas P mais on permutent b avec le vecteur de permutation puis on résoud par descente et remontée.

Voir exemple 3.12

ATTENTION P est en général le produit de plusieurs permutations

$$P = P_p P_{p-1} .. P_1 \Leftrightarrow P^{-1} = P_1 P_2 ... P_p = P^T$$

Dans le cas d'une matrice symétrique il est possible déconomiser de l'espace en imposant une condition supplémentaire à la décomposition LU. On impose que

$$U = L^T \Rightarrow A = LL^T$$

A étant symétrique on ne stocke que sa partie inférieure et sa diagonale. Alors on peut faire la même chose sur la décomposition et ne stocke que L.
La construction est basée sur Crout.

Algorithme de Choleski

- $L_{11} = (A_{11})^{\frac{1}{2}}$
- $L_{i1} = \frac{A_{i1}}{L_{ii}} \quad i = 2, ..., n$
- $k = 2, ..., n \; i = k+1, ..., n$
  - $L_{kk} = (A_{kk} - \sum_{j=1}^{k-1} L_{kj}^2)^{\frac{1}{2}}$
  - $L_{ik} = \frac{A_{ik} - \sum_{j=1}^{k-1} L_{ij} L_{kj}}{L_{ii}}$

Remarques

- Tout comme pour la décomposition LU on peut introduire une notation compacte et ”écraser“ A lors de la création de L
- La factorisation n'est pas unique : $A = (-L)(-L^T) = \tilde{L}\tilde{L}^T$ on rend la factorisation unique en imposant des pivots positifs : $L_{ii} > 0$.

Exemple 3.13

Utilisation de la factorisation de Choleski avec notation compacte pour

$$A = \begin{bmatrix} 4 & 6 & 2 \\ 6 & 10 & 5 \\ 2 & 5 & 14 \end{bmatrix}$$

Existence de la factorisation de Choleski

Soit

$$A = \begin{bmatrix} -1 & 2 \\ 2 & 6 \end{bmatrix}$$

On ne peut pas faire Choleski car $L_{11} = \sqrt{A_{11}}$ n'est pas réels.

En fait on aura pas existence de la factorisation si une des racines (les pivots) n'est pas réelles

Définition 3.5

Une matrice A est définie positive ssi

$$Ax \cdot x > 0 \quad \forall x \neq 0$$

Condition pour l'existence de la factorisation de Choleski

On peut montrer que

$$A \text{ symétrique définie positive} \Leftrightarrow A \text{ a une factorisation de Choleski}$$

La vérification de la positivité est difficile, la meilleur approche : appliquer la factorisation...

Dans le cas ou on n'a pas la positivité de la matrice on applique simplement la factorisation LU

- À partir de l'élimination de Gauss on a créé la méthode de décomposition LU
- U est la matrice obtenue par Gauss et L est composée de l'information jetée par Gauss pour transformer la matrice en triangulaire.
- Le système se résout maintenant avec une descente ($Ly = b$) suivie d'une remontée ($Ux = y$).
- la décomposition n'est pas unique
  - factorisation Doolittle (Matlab) : $L_{ii} = 1$
  - factorisation Crout : $U_{ii} = 1 \Leftarrow$ C'est elle que l'on retient
- Notation compacte : on écrase A en la remplaçant par $L + U - I$
- Les permutations sont exécutées après chaque construction d'une colonne de L et sont stockées dans un vecteur.

$$PA = LU$$

- En utilisant les propriétés des matrices triangulaires et de la factorisation le déterminant de la matrice A :

$$\det A = (\det P^T)(\det L)(\det U) = (\det P^T)(\det L)$$

Puisque $\det P^T = (-1)^p$ où p est le nombre de permutations effectuées on a

$$\det A = (-1)^p \prod_{i=1}^{n} L_{ii}$$

- Lors d'une factorisation le nombre d'opérations $\times/\div$ est de l'ordre de $n^3/3$
- Lors d'une remontée ou descente le nombre d'opérations $\times/\div$ est de l'ordre de n^2
- On cherche à optimiser encore l'algorithme : pour certains cas où la matrice est symétrique ($A = A^T$) on peut modifier la décomposition en imposant que $U = L^T$. C'est la factorisation de Choleski qui entraine une réduction en espace mémoire et un algorithme simplifié.

- La factorisation de Choleski est unique à un signe près. Pour la rendre unique on impose $L_{ii} > 0$
- La factorisation de Choleski existe a une condition (3 versions équivalentes) :
  - A est symétrique et définie positive : $A = A^T$ et $Ax \cdot x > 0 \quad \forall x \neq 0$.
  - Les déterminant de A et des sous-matrices principales de A sont tous strictement positifs.
  - Les valeurs propres de A sont toutes réelles et strictement positives.
- Dans le monde réel ($n > 10000$) la meilleure façon de vérifier qu'une factorisation de Choleski existe est de l'essayer. Les 3 conditions présentées ici sont plus couteuse (numériquement) que d'essayer Choleski et de ne pas réussir
- Si la factorisation de Choleski n'existe pas on fait un LU ordinaire ce qui implique que l'on ne profite pas de la structure particulière de A.

Exemple d'existence de la factorisation de Choleski

Soit

$$A = \begin{bmatrix} 4 & 6 & 2 \\ 6 & 10 & 5 \\ 2 & 5 & 14 \end{bmatrix}$$

On calcule les déterminants des sous-matrices principales.

Exemple d'existence de la factorisation de Choleski

Soit

$$A = \begin{bmatrix} -1 & 2 \\ 2 & 6 \end{bmatrix}$$

Alors $\det A_{11} = -1 < 0$ et la factorisation de Choleski n'existe pas.

Il y a une autre structure de matrice souvent rencontrée et pouvant mener à des simplification algorithmique et d'espace mémoire. Il s'agit de la matrice tridiagonales.

Définition 3.6

Une matrice est **tridiagonale** si ses seuls termes non nuls sont situés sur la diagonales principale et sur les deux diagonales adjacentes (au dessus et au dessous de la diago. principale)

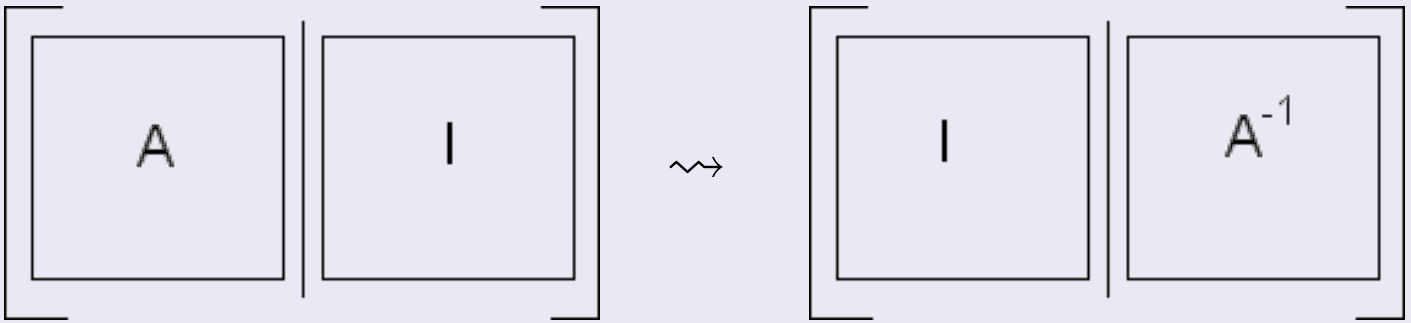
$$\begin{bmatrix} a_{11} & a_{12} & 0 & \dots & 0 \\ a_{21} & a_{22} & a_{23} & \ddots & 0 \\ 0 & a_{32} & a_{33} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & a_{n-1n} \\ 0 & 0 & 0 & a_{nn-1} & a_{nn} \end{bmatrix} = \begin{bmatrix} D_1 & S_1 & 0 & \dots & 0 \\ I_1 & D_2 & S_2 & \ddots & 0 \\ 0 & I_2 & D_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & S_{n-1} \\ 0 & 0 & 0 & I_{n-1} & D_n \end{bmatrix}$$

Concernant les tridiagonales :

- On stocke en mémoire les vecteurs D, I, S et b prenant au total seulement $4n - 2$ réels. On fait encore plus d'économie si la matrice est symétrique (on ignore S dans ce cas).
- L'algorithme de décomposition est beaucoup plus simple. On l'obtient en appliquant la décomposition de Crout pour ce cas particulier.
- On retrouve cette structure de matrice dans plusieurs applications des méthodes numériques :
  - différences finies (approximation de la solution d'équation différentielle ordinaire)
  - splines cubiques (approximation par une cubique d'une fonction à partir d'un groupe de points) connus

Comment inverser A ?

La méthode usuelle (i.e. à la main) consiste a augmenter la matrice avec la matrice identité et à faire une élimination jusqu'à ce qu'on ait une "identité à gauche"



En utilisant la décomposition LU cela revient à résoudre n fois le système $Ax = b$ avec $b = e_i$ (les vecteurs de la base euclidienne) pour i=1,...,n.
Clairement il ne faut pas utiliser cette approche pour résoudre un problème : on ferait n résolution et un produit matrice vecteur.

Nombre d'opérations $\times/\div$

Si on veut absolumment calculer l'inverse. Il faudra faire une factorisation LU suivi de n descente et de n remontée. Utilisant les ordres de grandeurs sur le nombre d'opérations $\times/\div$ on a

n	$Ax = b$	$A^{-1}b$
10	333	1 333
100	333 333	1 333 333
1000	333 333 333	1 333 333 333

Pour un système 1000×1000 on fait 10^9 opérations $\times/\div$ de plus pour résoudre le système.

1000 est un petit chiffre dans le monde réel...

Exemple 3.15

On se donne une matrice A obtenu par factorisation LU avec permutation :

$$A = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 0 & 0 \\ 3 & 0 & 1 \end{bmatrix} \quad L = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & -1/3 \end{bmatrix} \quad U = \begin{bmatrix} 1 & 0 & 1/3 \\ 0 & 1 & 1/2 \\ 0 & 0 & 1 \end{bmatrix}$$

Avec le vecteur de permutations

$$O = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}$$

On va calculer l'inverse. Il s'agit SURTOUT d'un exemple de résolution avec un vecteur de permutation

Exemple 3.16

Soit

$$A = \begin{bmatrix} 1.012 & -2.132 & 3.104 \\ -2.132 & 4.096 & -7.013 \\ 3.104 & -7.013 & 0.014 \end{bmatrix}$$

On veut appliquer la factorisation LU mais avec une mantisse fixe de 4 chiffres.

Cette restriction engendre des "perturbations" par rapport à la factorisation "humaine" (mantisse infinie). Quel est l'impact de cette perturbation lors de la résolution ?

Exemple 3.16 suite

Avec virgule flottante a 4 chiffres

$$L + U - I = \begin{bmatrix} 1.012 & -2.107 & 3.067 \\ -2.132 & -0.3960 & 1.197 \\ 3.104 & -0.4730 & -8.940 \end{bmatrix}$$

Avec virgule flottante humaine :

$$L + U - I = \begin{bmatrix} 1.012 & -2.106719... & 3.067193... \\ -2.132 & -0.395525... & 1.197757... \\ 3.104 & -0.473743... & -8.939140 \end{bmatrix}$$

Quel sont les effets sur la solution de $Ax = b$ pour b donné ?

Effet des opérations arithmétiques en virgules flottantes

On retourne aux considérations du chapitre 1. Pour le résoudre un système $Ax = b$ il est à prévoir que des erreurs de troncatures vont apparaitre.

On verra que pour certaines matrices les effets sur la solution seront négligeables. Pour certaines autres matrices les variations dans les entrées de la matrice induiront d'importantes erreurs dans la solution obtenues. De telles matrices sont dites **mal conditionnées**

Exemple 3.17

Soit une matrice A et une perturbation $\tilde{A}$ de A

$$A = \begin{bmatrix} 1. & 2 \\ 1.1 & 2 \end{bmatrix} \quad b = \begin{bmatrix} 10 \\ 10.4 \end{bmatrix} \quad \tilde{A} = \begin{bmatrix} 1. & 2 \\ 1.05 & 2 \end{bmatrix}$$

La solution exacte avec A est $(4, 3)$ et pour $\tilde{A}$ la solution est $(8, 1)$.
Une "petite" variation de A produit une "grande" variation de la solution.

Puisque l'arithmetique flottante (mantisse finie) mènent à de petites perturbations dans la décomposition LU. Il est possible que d'importants effets apparaissent sur la solution du système.

On peut produire d'importantes erreurs sur la solution numériques via la décomposition LU et le procèdé de descente et remontée.

Recettes simples pour éviter les dégats

Dépendant de la matrice A on pourra avoir ou non de large erreurs. On verra par la suite comment détecter de telle matrice. Pour le moment voici deux corrections simples à appliquer lors de la résolution pour éviter deux des principales causes d'erreurs :

- division par des nombres trop petits (pivots trop petits)
- la soustraction/addition avec des nombres ayant des ordres de grandeurs trop éloignées

Exemple 3.18

Soit

$$A = \begin{bmatrix} 0.0003 & 3.0000 \\ 1.0000 & 1.0000 \end{bmatrix} \quad b = \begin{bmatrix} 2.0001 \\ 1.0000 \end{bmatrix}$$

La solution est $x = [1/3, 2/3]^T$, en faisant la décomposition LU avec 4 chiffres on a

$$L+U-I = \begin{bmatrix} 0.3000 \times 10^{-3} & 0.1000 \times 10^{5} \\ 0.1000 \times 10^{1} & -0.9999 \times 10^{4} \end{bmatrix} \quad x = \begin{bmatrix} 0.6667 \times 10^{4} \\ 0.6667 \times 10^{0} \end{bmatrix}$$

La solution par LU est "loin" de la solution réelle. Comment éviter cette erreur ?

On fait une division "presque par zéro" à cause de A_{11} c'est lui qui modifie U_{12} et L_{22}. Alors on va éviter cette division en faisant une permutation.

Encore des permutations...

Soit

$$A = \begin{bmatrix} 1.0000 & 1.0000 \\ 0.0003 & 3.0000 \end{bmatrix} \quad b = \begin{bmatrix} 1.0000 \\ 2.0001 \end{bmatrix}$$

en faisant la décomposition LU avec 4 chiffres on a

$$L + U - I = \begin{bmatrix} 0.1000 \times 10^{1} & 0.1000 \times 10^{1} \\ 0.3000 \times 10^{-3} & 0.3000 \times 10^{1} \end{bmatrix} \quad x = \begin{bmatrix} 0.3333 \times 10^{0} \\ 0.6667 \times 10^{0} \end{bmatrix}$$

La solution par LU est "proche" de la solution réelle.

Recette 1

On peut éviter une partie des erreurs en faisant une permutation systèmatique des lignes lors de la décomposition : on s'assure d'avoir TOUJOURS le pivot de plus grand en valeur absolue sur la diagonale.

Exemple 3.19

$$A = \begin{bmatrix} 1 & 6 & 9 \\ 2 & 1 & 2 \\ 3 & 6 & 9 \end{bmatrix}$$

Exemple 3.20

$$A = \begin{bmatrix} 2.0000 & 100000 \\ 1.0000 & 1.0000 \end{bmatrix} \quad b = \begin{bmatrix} 100000 \\ 2.0000 \end{bmatrix}$$

La solution est $x = [1.00002, 0.99998]^T$, en faisant la décomposition LU avec 4 chiffres on a

$$L+U-I = \begin{bmatrix} 0.2000 \times 10^1 & 0.5000 \times 10^5 \\ 0.1000 \times 10^1 & -0.5000 \times 10^5 \end{bmatrix} \quad x = \begin{bmatrix} 0.0000 \times 10^0 \\ 0.1000 \times 10^1 \end{bmatrix}$$

La solution par LU est "loin" de la solution réelle. Comment éviter cette erreur ?

En calculant L_{22} on fait une soustraction de chiffres d'ordre de grandeur très différents (opérations à éviter Chap 1) à cause de A_{12} . Alors on va éviter cette opérations en multipliant la ligne par une constante : utilisation d'une transformation M.

Matrice M

On divise la première ligne de A par 100000 ce qui donne

$$A = \begin{bmatrix} 0.2000 \times 10^{-4} & 0.1000 \times 10^{1} \\ 0.1000 \times 10^{1} & 0.1000 \times 10^{1} \end{bmatrix} \quad b = \begin{bmatrix} 0.1000 \times 10^{1} \\ 0.2000 \times 10^{1} \end{bmatrix}$$

en faisant la décomposition LU avec 4 chiffres AVEC permutation on a

$$L + U - I = \begin{bmatrix} 0.1000 \times 10^{1} & 0.1000 \times 10^{1} \\ 0.2000 \times 10^{-4} & 0.1000 \times 10^{1} \end{bmatrix} \quad x = \begin{bmatrix} 0.1000 \times 10^{1} \\ 0.1000 \times 10^{1} \end{bmatrix}$$

La solution par LU est "proche" de la solution réelle.

Definition 3.7

Une **mise à l'échelle d'une matrice A** consiste à diviser chaque ligne de A par le plus grand termes en valeur absolue de la ligne correspondante.

Recette 2

On peut éviter une partie des erreurs en faisant une mise à l'échelle systèmatique d'une matrice avant la décomposition.

Attention au déterminant

L'introduction de matrice M à un effet important sur le déterminant. Si on multiplie une matrice par un scalaire on récupère son déterminant en divisant le déterminant de L par ce scalaire.

On veut étudier le comportement de la solution calculée par rapport à la solution exacte pour un système linéaire. On va d'abord introduite une mesure de la distance dans l'espace $\mathbb{R}^n$.

Definition 3.8

Une **norme vectorielle** est une application de $\mathbb{R}^n$ dans $\mathbb{R}$ notée $\|\|$ vérifiant

1. Positivité stricte :

$$\|x\| > 0 \quad \forall x \in \mathbb{R}^n, x \neq 0$$

2. $\forall \alpha \in \mathbb{R}$

$$\|\alpha x\| = |\alpha| \|x\| \quad \forall x \in \mathbb{R}^n$$

3. Inégalité du triangle

$$\|x + y\| \leq \|x\| + \|y\| \quad \forall x, y \in \mathbb{R}^n$$

.

Definition 3.9

La norme euclidienne d'un vecteur x notée $\|x\|_2$ est définie par

$$\|x\|_2 = \left(x_1^2 + x_2^2 + ... + x_n^2\right)^{\frac{1}{2}}$$

évidemment la norme euclidienne est une norme vectoriel.

Definition 3.10

Il existe d'autre norme sur $\mathbb{R}^n$.

- La norme l_1 : $\|x\|_1 = \sum_{i=1}^{n} |x_i|$
- La norme l_∞ : $\|x\|_\infty = \max_{i=1}^{n} |x_i|$

Definition 3.11

On peut aussi introduire une norme matricielle. On l'a définie comme une application $A :\rightarrow \|A\|$ vérifiant

1. Positivité stricte :
$$\|A\| > 0 \quad \forall A, A \neq 0$$
2. $\forall \alpha \in \mathbb{R}$
$$\|\alpha A\| = |\alpha| \|A\| \quad \forall A$$
3. Inégalité du triangle
$$\|A + B\| \leq \|A\| + \|B\| \quad \forall A, B \text{de dimensions compatibles}$$
4.
$$\|AB\| \leq \|A\| \|B\| \quad \forall A, B \text{de dimensions compatibles}$$

Une norme matricielle DOIT vérifier 1-4.

Norme induite

On peut construire une norme matricielle à partir d'une norme vectorielle :

$$\|A\| = \sup_{\|x\|=1} \|Ax\|$$

Une telle norme est dite **norme matricielle induite par la norme vectorielle**. On dit aussi que cette norme est **subordonnée** à la norme vectorielle

Toute les normes ne sont pas induite

La norme de Frobenius définie par

$$\|A\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij}^2 \right)^{\frac{1}{2}}$$

est une norme matricielle, s'apparentant à la norme euclidienne mais elle n'est induite par aucune norme.

Théorème 3.5

Les normes induites par les normes l_1 et l_∞ sont

- $\|A\|_1 = \sup_{\|x\|_1=1} \|Ax\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{n} |A_{ij}|$
- $\|A\|_\infty = \sup_{\|x\|_\infty=1} \|Ax\|_\infty = \max_{1\leq i\leq n} \sum_{j=1}^{n} |A_{ij}|$

La norme $\|A\|_1$ consiste à trouver la colonne dont la somme des éléments (en valeur absolue) est la plus grande.

La norme $\|A\|_\infty$ consiste à trouver la ligne dont la somme des éléments (en valeur absolue) est la plus grande.

Norme induite par la norme euclidienne

La norme induite par la norme euclidienne est définie par

$$\|A\|_2 = \sup_{\|x\|_2=1} \|Ax\|_2 = \left(\rho(AA^T)\right)^{\frac{1}{2}}$$

où ρ est la plus grande valeur propre (en valeur absolue) de la matrice AA^T.

Pourquoi s'intéresser au norme matricielle ? On va manipuler des vecteurs résultants de produit matrice vecteur. Ce qui nous amène à une définition fort importante

Définition 3.12

On dit que la norme matricielle et la norme vectorielle sont compatibles si

$$\|Ax\| \leq \|A\|\|x\|$$

pour toutes matrice A et vecteurs x de dimensions cohérentes.

Attention au mélange de normes : à gauche norme vectorielle, à droite matricielle et vectorielle (respectivement).

Normes compatibles

On peut montrer que

- $\|Ax\|_1 \leq \|A\|_1 \|x\|_1$ la norme 1 des matrices et vecteurs est compatible
- $\|Ax\|_\infty \leq \|A\|_\infty \|x\|_\infty$ la norme ∞ des matrices et vecteurs est compatible
- $\|Ax\|_2 \leq \|A\|_F \|x\|_2$ la norme Frobenius des matrices est compatible avec la norme euclidienne.

La dernière relation montre qu'il n'y a pas de lien entre norme induite et norme compatible.

Soit le système $Ax = b$ et x^* une solution obtenue par arithmétique flottante. On veut étudier l'erreur par rapport à la solution exacte :

$$e = x - x^*$$

Si tout va bien $\|e\| = \|x - x^*\|$ est petit.
Posons $r = b - Ax^*$ r est le résidu de notre résolution.

$$r = b - Ax^* = Ax - Ax^* = A(x - x^*) = Ae$$

On cherche à avoir une borne inférieure et supérieure sur $\|e\|$. On va se servir de la compatibilité des normes (on choisi une norme compatible)

$$A^{-1}r = e \Rightarrow \|e\| = \|A^{-1}r\| \leq \|A^{-1}\|\|r\|$$

On a aussi

$$\|r\| = \|Ae\| \leq \|A\|\|e\|$$

Ce qui nous donne

$$\frac{\|r\|}{\|A\|} \leq \|e\| \leq \|A^{-1}\|\|r\|$$

$$\frac{\|r\|}{\|A\|} \leq \|e\| \leq \|A^{-1}\|\|r\|$$

On veut faire apparaitre x dans cette relation (on voudrait l'"erreur relative") alors on revient au système :

$$\|b\| = \|Ax\| \leq \|A\|\|x\| \Rightarrow \frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|}$$

$$\|x\| = \|A^{-1}b\| \leq \|A^{-1}\|\|b\| \Rightarrow \frac{1}{\|A^{-1}\|\|b\|} \leq \frac{1}{\|x\|}$$

En combinant ces deux inégalités on a

$$\frac{1}{\|A^{-1}\|\|b\|} \leq \frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|}$$

On multiplie la dernière relation par $\|e\|$

$$\frac{\|e\|}{\|A^{-1}\|\|b\|} \leq \frac{\|e\|}{\|x\|} \leq \frac{\|A\|}{\|b\|}\|e\|$$

et on a

$$\frac{\|r\|}{\|A\|\|A^{-1}\|\|b\|} \leq \frac{\|e\|}{\|x\|} \leq \frac{\|A^{-1}\|\|A\|\|r\|}{\|b\|}$$

Definition 3.13

Le conditionnement d'une matrice A noté condA est défini par

$$condA = \|A^{-1}\|\|A\|$$

Remarque

- condA dépend de la norme matricielle utilisée
- Puisque $1 \leq \|I\|$ parce que

$$\|A\| = \|AI\| \leq \|A\|\|I\|$$

on a

$$1 \leq \|I\| \leq \|AA^{-1} \leq \|A^{-1}\|\|A\|$$

et

$$1 \leq condA < \infty$$

Théorème 3.6

$$\frac{1}{condA}\frac{\|r\|}{\|b\|} \leq \frac{\|e\|}{\|x\|} \leq CondA\frac{\|r\|}{\|b\|}$$

Concernant le conditionnement

$$\frac{1}{condA}\frac{\|r\|}{\|b\|} \leq \frac{\|e\|}{\|x\|} \leq CondA\frac{\|r\|}{\|b\|}$$

1. le terme du milieu représente l'erreur relative
2. Si condA est près de 1 alors

$$\frac{\|e\|}{\|x\|} \approx \frac{\|r\|}{\|b\|}$$

   est la solution est bonne si $\|r\|$ est petit.
3. si condA est grand alors l'erreur relative peut être grande.
4. même si $\|r\|$ est petit il est possible que l'erreur soit grande (manque le conditionnement petit)
5. Plus condA augmente plus l'algorithme de résolution doit être précis (pour réduire $\|r\|$)

Concernant le conditionnement

$$\frac{1}{condA}\frac{\|r\|}{\|b\|} \leq \frac{\|e\|}{\|x\|} \leq CondA\frac{\|r\|}{\|b\|}$$

1. Même si condA est petit si la résolution est imprécise on pourra avoir des résultats erronés (manque $\|r\|$ est petit).
2. On peut montrer que

$$\max\left(\frac{\|e\|\|b\|}{\|x\|\|r\|}, \frac{\|r\|\|x\|}{\|e\|\|b\|}\right) \leq condA$$

3. condA dépend de la norme matricielle choisie (géneralement la norme ∞) mais ne dépend pas de l'algorithme de résolution du système. C'est une mesure de la sensibilité de la matrice au perturbation lors de manipulations matricielles.

Erreurs dans A

On veut résoudre le système $Ax = b$ mais à cause d'erreur de troncature ou dans la mesure des éléments de A on résout plutôt le système

$$(A + E)x = b$$

Soit x^* solution du système perturbé et x la solution du système original.

$$x = A^{-1}b = A^{-1}((A + E)x^*) = (I + A^{-1}E)x^* = x^* + A^{-1}Ex^*$$

Donc si les normes sont compatibles

$$\|e\| = \|x - x^*\| = \|A^{-1}Ex^*\| \leq \|A^{-1}\|\|E\|\|x^*\|$$

Et on a

$$\frac{\|x - x^*\|}{\|x^*\|} \leq condA \frac{\|E\|}{\|A\|}$$

Erreurs de troncature sur A

$$\frac{\|x - x^*\|}{\|x^*\|} \leq condA \frac{\|E\|}{\|A\|}$$

- $\frac{\|x-x^*\|}{\|x^*\|}$ approxime l'erreur relative
- $\frac{\|E\|}{\|A\|}$ ressemble à l'erreur relative sur les entrées de la matrice A
- Si condA est petit et $\|E\|$ est petit alors l'erreur relative sera petite : bonne approximation de x
- Si condA est grand on ne peut rien dire

Erreurs dans A

$$\frac{\|x - x^*\|}{\|x^*\|} \leq condA \frac{\|E\|}{\|A\|}$$

- Si E représente l'erreur de troncature sur une machine alors

$$|E_{ij}| \leq \epsilon_m |A_{ij}|$$

par définition de la troncature donc

$$\sum_j |E_{ij}| \leq \epsilon_m \sum_j |A_{ij}| \Leftrightarrow \|E\|_\infty \leq \epsilon_m \|A\|_\infty$$

d'où

$$\frac{\|x - x^*\|}{\|x^*\|} \leq \epsilon_m condA$$

Si le conditionnement est mauvais il faut plus de précision sur les nombres

Quoi faire de condA ?

On c'est donné cette mesure qu'en fait-on ?

- Exemple 3.16 (3.24) Le conditionnement de A = 270.51 correspond à une matrice sensible au perturbations. Ne veut pas dire que je n'aurai pas la bonne solution mais que j'ai de bonne chance d'obtenir une solution erronnées sans un bon algorithme
- Exemple 3.18 (3.26) condA = 4 ce qui est faible indiquant que si tout va bien j'aurai une bonne solution. Rappelons que sans permutations la solution est mauvaise ce qui ne contredit pas le conditionnement mais indique que mon algorithme est mauvais. En incluant les permutations on obtient une bonne solution.
- $$A = \begin{bmatrix} 1. & 2 \\ 1.1 & 2 \end{bmatrix} \quad A^{-1} = \begin{bmatrix} -10. & 10. \\ -5.5 & 5. \end{bmatrix}$$

  cond A = 62 indiquant un mauvais conditionnement.

Quoi faire de condA ?

- Est-ce que je peux diminuer le conditionnement en changeant de norme ? Non, on a "équivalence des normes" faisant en sorte que rien ne changera en prenant d'autre norme.
- Alors quoi ?
  - Mauvais conditionnement == appliquer les recettes 1 et 2 OBLIGATOIREMENT et être critique des résultats obtenus et choisir avec soin l'algorithme de résolution (contrôler la qualité autrement, de manière ad hoc)
  - Bon conditionnement == les recettes ne sont pas nécessaires, on peut croire que tout va bien mais on doit encore être critique des résultats obtenus (contrôler la qualité autrement, de manière ad hoc).

- Décomposition LU
- Structures particulières menant à des algo simplifiés et plus efficaces :
  - A symétrique, définie poisitive : Choleski
  - A tridiagonale
- Effets de la représentation virgule flottante
  - Recette 1 : permutation des lignes après chaque construction de colonne de L
  - Recette 2 : mise à l'échelle des lignes de la matrice
  - Conditionnement : un indicateur de la sensibilité du système lors de la résolution
  - $CondA + r = (Ax - b)$ donne un verdict positif si les deux sont petits
  - $CondA$ seul = rien
  - $r = Ax - b$ seul = rien

Exemple introductif

Supposons que nous cherchions à résoudre le système suivant :

$$\begin{cases} 2x_1 & +4x_2 & -2x_3 & -6x_4 & = -6 \\ 1x_1 & +3x_2 & & -6x_4 & = 0 \\ 3x_1 & -x_2 & +x_3 & -6x_4 & = 8 \\ & -x_2 & +2x_3 & -6x_4 & = 6 \end{cases}$$

et que nous connaissions un vecteur $x^*_1, x^*_2, x^*_3, x^*_4$ proche de la solution.

$$\begin{cases} x_1 = \dfrac{-6 - 4x^*_2 + 2x^*_3 + 6x^*_4}{2} \\ x_2 = \dfrac{0 - 1x^*_1 - 0x^*_3 + 6x^*_4}{3} \\ x_3 = \dfrac{8 - 4x^*_1 + x^*_2 + 6x^*_4}{1} \\ x_4 = \dfrac{6 - 4x^*_1 + x^*_2 - 2x^*_3}{-6} \end{cases}$$

La politique des « petits » pas...

on cherche à créer une suite de vecteurs $\{x^{(k)}, k \in \mathbb{N}\}$ qui converge vers $\widetilde{x}$ solution de $A\widetilde{x} = \mathrm{b}$

- Jacobi $x_J^{(k)}, k \in \mathbb{N}$
- Gauss Seidel $x_{GS}^{(k)}, k \in \mathbb{N}$
- Relaxation $x_R^{(k)}, k \in \mathbb{N}$

a chaque itération...

le vecteur d'erreurs : $e^{(k)} = x^{(k)} - \widetilde{x}$ doit diminuer

comment choisir ?

- il en existe plein d'autres...
- la question centrale : quand convergent'elles ?
- seconde question : à quelle vitesse

parmi celles qui convergent on prend celle qui converge le plus vite !

$$Ax = b \iff \sum_{j=1}^{i-1} A_{ij}x_j + A_{ii}x_i + \sum_{j=i+1}^{n} A_{ij}x_j = b_i \qquad i = 1, n$$

Fonction $x \leftarrow \texttt{Jacobi}(A, b, x)$

Tant que (on a pas convergé) **faire**
  Pour $i = 1, n$ **faire**
    $$y_i = \frac{b_i - \sum_{j=1}^{i-1} A_{ij}x_j - \sum_{j=i+1}^{n} A_{ij}x_j}{A_{ii}}$$
  Fin Pour
  Pour $i = 1, n$ **faire**
    $x_i = y_i$
  Fin Pour
Fait

Exemple : méthode de Jacobi

$$A = \begin{pmatrix} 3 & 1 & -1 \\ 1 & 2 & 0 \\ -1 & 1 & 4 \end{pmatrix} \quad \text{avec } b = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad \text{et } \mathrm{x}_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \mathrm{x}_1 = \begin{pmatrix} 1/3 \\ 1/2 \\ 1/4 \end{pmatrix}$$

- $y_1 = \frac{1-1/2+1/4}{3} = \frac{3}{12}$
- $y_2 = \frac{1-1/3}{2} = \frac{1}{3}$
- $y_3 = \frac{1+1/3-1/2}{4} = \frac{5}{24}$

$$\mathrm{x}_2 = \begin{pmatrix} 3/12 \\ 1/3 \\ 5/24 \end{pmatrix}$$

$$\mathrm{norm}(A * \mathrm{x}_0 - \mathrm{b}) = 1.7321$$
$$\mathrm{norm}(A * \mathrm{x}_1 - \mathrm{b}) = 0.4488$$
$$\mathrm{norm}(A * \mathrm{x}_2 - \mathrm{b}) = 0.1718$$

$$[\mathrm{x}_2 \quad A\backslash \mathrm{b}] = \begin{pmatrix} 0.2500 & 0.2941 \\ 0.3333 & 0.3529 \\ 0.2083 & 0.2353 \end{pmatrix}$$

Les itérations de Gauss Seidel

Fonction $\mathrm{x} \leftarrow$ `Gauss_Seidel`$(A, \mathrm{b}, \mathrm{x})$

Tant que (on a pas convergé) **faire**
  Pour $i = 1, n$ **faire**
    $$x_i = \frac{b_i - \sum_{j=1}^{i-1} A_{ij} x_j - \sum_{j=i+1}^{n} A_{ij} x_j}{A_{ii}} ;$$
  Fin Pour
Fait

plus « simple », mais non parallélisable

Les itérations de relaxation

Fonction $\mathrm{x} \leftarrow$ `Relaxation`$(A, \mathrm{b}, \mathrm{x}, \omega)$

Tant que (on a pas convergé) **faire**
  Pour $i = 1, n$ **faire**

$$x_i = \omega \left(\frac{b_i - \sum_{j=1}^{i-1} A_{ij} x_j - \sum_{j=i+1}^{n} A_{ij} x_j}{A_{ii}} \right) + (1 - \omega) x_i$$

  Fin Pour
Fait

pour $\omega = 1$ c'est Gauss Seidel

Quand s'arrêter ?

- $\|\mathrm{x}^{\text{new}} - \mathrm{x}^{\text{old}}\| < \varepsilon$
- un nombre maximal d'itérations est atteint

Fonction $\mathrm{y} \leftarrow$ `Gauss_Seidel`$(A, \mathrm{b}, \mathrm{x}, \varepsilon, n\_\text{ite\_max})$
Pour $i = 1, n$ **faire**
  $y_i = x_i \quad x_i = y_i + 2 * \varepsilon$
Fin Pour
$n\_\text{ite} = 0$
Tant que ($\|\mathrm{x} - \mathrm{y}\| > \varepsilon$ ET $n\_\text{ite} < n\_\text{ite\_max}$) **faire**
  Pour $i = 1, n$ **faire**
    $x_i = y_i$
  Fin Pour
  Pour $i = 1, n$ **faire**
    $y_i = \dfrac{b_i - \sum_{j=1}^{i-1} A_{ij} y_j - \sum_{j=i+1}^{n} A_{ij} x_j}{A_{ii}}$
  Fin Pour
  $n\_\text{ite} = n\_\text{ite} + 1$
Fait

- $\mathrm{x} = \mathrm{x}^{\text{old}}$
- $\mathrm{y} = \mathrm{x}^{\text{new}}$

Formulation matricielle des itérations

Considérons maintenant la suite de vecteurs $\mathrm{x}^{(k)}$ construit à l'aide des matrices carrées M et N, ainsi que d'un vecteur b :

$$M\mathrm{x}^{(k+1)} = N\mathrm{x}^{(k)} + \mathrm{b}$$

On peut réécrire ces itérations à l'aide de la matrice carrée $C = M^{-1}N$ et du vecteur $\mathrm{d} = M^{-1}\mathrm{b}$:

$$\begin{array}{rcccccc} \mathrm{x}^{(k+1)} & = & M^{-1}N & \mathrm{x}^{(k)} & + & M^{-1}\mathrm{b} \\ & = & C & \mathrm{x}^{(k)} & + & \mathrm{d} \end{array}$$

pour un vecteur $x^{(0)}$ donné.

Décomposition d'une matrice

Revenons au système $A\mathrm{x} = \mathrm{b}$. Nous pouvons décomposer la matrice A

$$\underset{A}{\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}} = \underset{D}{\begin{pmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{pmatrix}} + \underset{U}{\begin{pmatrix} 0 & a_{12} & a_{13} \\ 0 & 0 & a_{23} \\ 0 & 0 & 0 \end{pmatrix}} + \underset{L}{\begin{pmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ a_{31} & a_{32} & 0 \end{pmatrix}}$$

diagonale — triangulaire sup — triangulaire inf

$$\underset{A\mathrm{x}}{\begin{pmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \end{pmatrix}} = \underset{D\mathrm{x}}{\begin{pmatrix} a_{11}x_1 & +0 & +0 \\ 0 & +a_{22}x_2 & +0 \\ 0 & +0 & +a_{33}x_3 \end{pmatrix}} + \underset{(L+U)\mathrm{x}}{\begin{pmatrix} 0 & +a_{12}x_2 & +a_{13}x_3 \\ a_{21}x_1 & +0 & +a_{23}x_3 \\ a_{31}x_1 & +a_{32}x_2 & +0 \end{pmatrix}}$$

$$A\mathrm{x} = D\mathrm{x} + L\mathrm{x} + U\mathrm{x}$$

Décomposition d'une matrice

Revenons au système $A\mathrm{x} = \mathrm{b}$. Nous pouvons décomposer la matrice A

De manière générale : $A = D + L + U$ avec :

$$D = diag(a_{11}, a_{22}, ..., a_{nn})$$

$$L = \begin{pmatrix} 0 & 0 & ... & ... & 0 \\ a_{21} & 0 & ... & ... & 0 \\ a_{31} & a_{32} & ... & ... & 0 \\ ... & & & 0 & 0 \\ a_{n1} & a_{n2} & ... & a_{n,n-1} & 0 \end{pmatrix}$$

$$U = \begin{pmatrix} 0 & a_{12} & a_{13} & ... & a_{1n} \\ 0 & 0 & ... & ... & a_{2n} \\ 0 & & ... & ... & \\ ... & & & a_{n-2,n-1} & a_{n-2,n} \\ ... & & & 0 & a_{n-1,n} \\ 0 & 0 & ... & 0 & 0 \end{pmatrix}$$

Les itérations de Jacobi s'écrivent alors

Le point fixe

$$\begin{aligned} A\mathrm{x} &= \mathrm{b} \\ (D+L+U)\mathrm{x} &= \mathrm{b} \\ D\mathrm{x}+(L+U)\mathrm{x} &= \mathrm{b} \\ D\mathrm{x} &= -(L+U)\mathrm{x} + \mathrm{b} \end{aligned}$$

Les itérations de Jacobi

$$D\mathrm{x}^{(k+1)} = -(U+L)\mathrm{x}^{(k)} + \mathrm{b}$$

$$\mathrm{x}^{(k+1)} = \underbrace{-D^{-1}(U+L)}_{C_j}\mathrm{x}^{(k)} + \underbrace{D^{-1}\mathrm{b}}_{\mathrm{d}_j}$$

Les itérations de Jacobi s'écrivent alors

Fonction $\mathrm{x} \leftarrow \texttt{Jacobi}(A, \mathrm{b}, \mathrm{x})$

Tant que (on a pas convergé) **faire**
  Pour $i = 1, n$ **faire**
    $$y_i = \frac{b_i - \sum_{j=1}^{i-1} A_{ij} x_j - \sum_{j=i+1}^{n} A_{ij} x_j}{A_{ii}}$$
  Fin Pour
  Pour $i = 1, n$ **faire**
    $x_i = y_i$
  Fin Pour
Fait

Fonction $\mathrm{x} \leftarrow \texttt{Jacobi}(A, \mathrm{b}, \mathrm{x})$

Tant que (on a pas convergé) **faire**
  $\mathrm{x} = D \backslash (b - (L + U)\mathrm{x})$
Fait

$$\mathrm{x}^{(k+1)} = \underbrace{C_j}_{-D^{-1}(U+L)} \mathrm{x}^{(k)} + \underbrace{\mathrm{d}_j}_{D^{-1}\mathrm{b} = D \backslash b}$$

Gauss Seidel et la relaxation

les itérations de Gauss Seidel s'écrivent

$$\begin{aligned}(D+L)\mathrm{x}^{(k+1)} &= -U\mathrm{x}^{(k)} + \mathrm{b} \\ \mathrm{x}^{(k+1)} &= \underbrace{(D+L)^{-1}(-U)}_{C_g}\mathrm{x}^{(k)} + \underbrace{(D+L)^{-1}\mathrm{b}}_{\mathrm{d}_g}\end{aligned}$$

et les itérations de relaxation s'écrivent

$$\begin{aligned}(D+\omega L)\mathrm{x}^{(k+1)} &= \big((1-\omega)D - \omega U\big)\mathrm{x}^{(k)} + \omega\mathrm{b} \\ \mathrm{x}^{(k+1)} &= \underbrace{\big((D+\omega L)\big)^{-1}\big((1-\omega)D - \omega U\big)}_{C_r}\mathrm{x}^{(k)} + \\ &\qquad \underbrace{\big((D+\omega L)\big)^{-1}(\omega\mathrm{b})}_{\mathrm{d}_r}\end{aligned}$$

Pour $w = 1$, on retrouve bien l'équivalence des formulations de Gauss Seidel et de la relaxation

Condition suffisante de convergence

$$\mathrm{x}^{(k+1)} = C\mathrm{x}^{(k)} + \mathrm{d}$$

Si $\widetilde{x}$ est solution du problème, on a $\widetilde{x} = C\widetilde{x} + \mathrm{d}$ et donc

$$\begin{aligned} \mathrm{x}^{(k+1)} - \widetilde{x} &= C(\mathrm{x}^{(k)} - \widetilde{x}) \\ &= C^2(\mathrm{x}^{(k-1)} - \widetilde{x}) \\ &\dots \\ &= C^{k+1}(\mathrm{x}^{(0)} - \widetilde{x}) \end{aligned}$$

Si maintenant on raisonne en norme :

$$\overbrace{\|\mathrm{x}^{(k+1)} - \widetilde{x}\|}^{\text{erreur à l'étape } k+1} = \|C^{k+1}(\mathrm{x}^{(0)} - \widetilde{x})\|$$
$$\leq \|C^{k+1}\|\|\mathrm{x}^{(0)} - \widetilde{x}\| \leq \underbrace{\|C\|^{k+1}}_{\to 0\ ?} \underbrace{\|\mathrm{x}^{(0)} - \widetilde{x}\|}_{\text{erreur initiale}}$$

ce faisant, nous avons démontré le résultat suivant :

Theorem (convergence d'une suite vectorielle)

S'il existe une norme matricielle telle que $\|C\| < 1$ *alors la suite* $\mathrm{x}^{(k)}$ *converge vers* $\widetilde{x} = (I - C)^{-1}d$

Normes matricielles de type vectorielle

Norme vectorielle (norme de Frobenius)

$$\|C\|_F^2 = \sum_i \sum_j C_{ij}^2$$

Ce type de norme n'est pas toujours pertinent pour ce que nous cherchons à faire. En effet, certaines normes vectorielles ne sont pas sous multiplicatives (la propriété $\|AB\| \leq \|A\|\|B\|$ n'est pas toujours vérifié) ce qui est fâcheux pour notre analyse.
Par exemple pour la norme vectorielle $\|A\|_\Delta = \max_{i,j} |A_{ij}|$:

$$A = B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \qquad \|AB\|_\Delta = 2 \not\leq \|A\|_\Delta \|B\|_\Delta = 1$$

Normes matricielles d'opérateur

Pour conserver cette propriété, il faut utiliser une *Norme d'opérateur* définie par :

$$\|C\| = \sup_{x \neq 0} \frac{\|Cx\|}{\|x\|}$$

Par construction on a bien :

$$\|ABx\| \leq \|A\|\|Bx\| \leq \|A\|\|B\|\|x\| \quad \Rightarrow \quad \|AB\| \leq \|A\|\|B\|$$

Exemples (théorèmes) :

- $\|C\|_1 = \max_j \sum_i |C_{ij}|$
- $\|C\|_\infty = \max_i \sum_j |C_{ij}|$
- $\|C\|_2 =$
  - $\max_i \sqrt{\mu_i}$ où μ_i est valeur propre de $C^\top C$.
  - pour C symétrique : $\max_i |\lambda_i|$ $\lambda_i = \sqrt{\mu_i}$ est val. propre de C.

démonstrations : de manière générale nous allons montrer les deux inégalités suivantes

$$\phi(C) \leq \|C\| \leq \phi(C)$$

On commence par trouver une borne sur cette norme :

$$\begin{aligned}
\|C\|_1 &= \sup_{\mathrm{x}\neq 0} \frac{\|C\mathrm{x}\|}{\|\mathrm{x}\|} \\
&= \sup_{\mathrm{x}\neq 0} \frac{\sum_i \left|\sum_j C_{ij} x_j\right|}{\|\mathrm{x}\|} \\
&\leq \sup_{\mathrm{x}\neq 0} \frac{\sum_i \sum_j |C_{ij}||x_j|}{\|\mathrm{x}\|} \\
&\leq \sup_{\mathrm{x}\neq 0} \sum_j \Big(\sum_i |C_{ij}|\Big) \frac{|x_j|}{\|\mathrm{x}\|} \\
&\leq \sup_{\mathrm{x}\neq 0} \sum_j \Big(\max_j \sum_i |C_{ij}|\Big) \frac{|x_j|}{\|\mathrm{x}\|} \\
&\leq \sup_{\mathrm{x}\neq 0} \Big(\max_j \sum_i |C_{ij}|\Big) \sum_j \frac{|x_j|}{\|\mathrm{x}\|} \\
&\leq \max_j \sum_i |C_{ij}| \qquad \text{car } \frac{|x_j|}{\|\mathrm{x}\|} \leq 1
\end{aligned}$$

en choisissant le vecteur $\mathrm{e}_j = (0, \ldots, 0, 1, 0, \ldots, 0)^\top$, on atteint la borne

Le cas de la norme 2

on passe par le système des vecteurs propres : $C^\top C \mathrm{v}_i = \mu_i \mathrm{v}_i$

$$\mathrm{x} = \sum_i \xi_i \mathrm{v}_i \quad \Leftrightarrow \quad \mathrm{x} = P\xi \qquad ; \qquad \|\mathrm{x}\|^2 = \xi^\top P^\top P \xi = \sum_i \xi_i^2$$

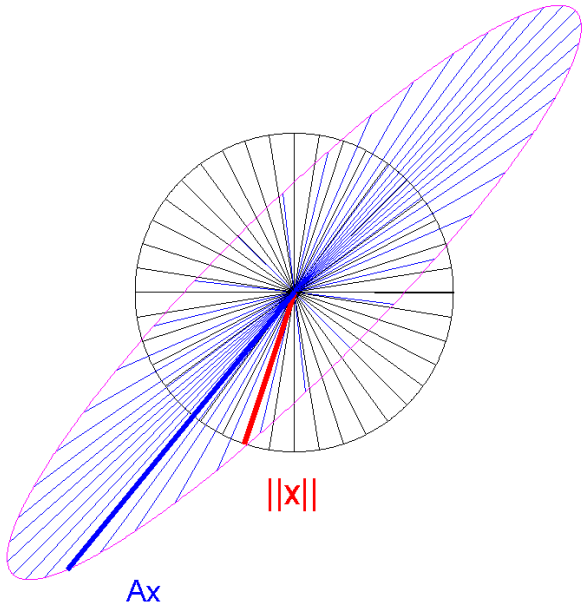
$$\begin{aligned}
\|C\|_2^2 &= \sup_{\mathrm{x}\neq 0} \frac{\|C\mathrm{x}\|^2}{\|\mathrm{x}\|^2} \\
&= \sup_{\mathrm{x}\neq 0} \frac{\mathrm{x}^\top C^\top C \mathrm{x}}{\|\mathrm{x}\|^2} \\
&= \sup_{\mathrm{x}\neq 0} \frac{\mathrm{x}^\top C^\top C \left(\sum_i \xi_i \mathrm{v}_i\right)}{\|\mathrm{x}\|^2} \\
&= \sup_{\mathrm{x}\neq 0} \frac{\mathrm{x}^\top \left(\sum_i \xi_i \mu_i \mathrm{v}_i\right)}{\|\mathrm{x}\|^2} \\
&= \sup_{\mathrm{x}\neq 0} \frac{\sum_i \xi_i^2 \mu_i}{\|\mathrm{x}\|^2} \qquad \leq \max_i \mu_i
\end{aligned}$$

Là encore, en choisissant le vecteur propre associé, on atteint la borne :

$$\frac{\|C\mathrm{v}_i\|^2}{\|\mathrm{v}_i\|^2} = \mathrm{v}_i C^\top C \mathrm{v}_i = \mu_i$$

Illustration 2d...

$$\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \sup_{\|x\|=1} \|Ax\|$$



Definition

rayon spectral

$$\rho(A) = \max_{1 \leq i \leq n} |\lambda_i|$$

Propriétés des normes (Op)

- $\|A\| > 0$ si $A \neq 0$
- $\|A\| = 0 \Leftrightarrow A = 0$
- $\|kA\| = |k|\|A\|$
- $\|A + B\| \leq \|A\| + \|B\|$
- $\|AB\| \leq \|A\|\|B\|$

Propriétés des normes

- $\|A\|_1 = \max_j \sum_i |A_{ij}|$
- $\|A\|_\infty = \max_i \sum_j |A_{ij}|$
- A sym. : $\|A\|_2 = \rho(A)$
- A sym. : $\rho(A) \leq \|A\|$
- $\|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty}$

Résumons nous (normes matricielles ?)

1. on cherche à résoudre $A\mathrm{x} = \mathrm{b}$
2. on utilise une méthode itérative : $\mathrm{x}^{(k+1)} = C\mathrm{x}^{(k)} + \mathrm{d}$
3. l'erreur... $\mathrm{e}^{(k)} = C^k \mathrm{e}^{(0)}$
4. ... peut être contrôlée : $\|\mathrm{e}^{(k)}\| \leq \|C\|^k \|\mathrm{e}^{(0)}\|$
5. si l'on trouve une norme telle que $\|C\| < 1$ la suite converge
6. définir une norme telle que $\|C^k\| \leq \|C\|^k$
7. définition : $\|C\| = \sup_{\mathrm{x}\neq 0} \frac{\|C\mathrm{x}\|}{\|\mathrm{x}\|}$
8. théorèmes (bien pratiques...) :
   - $\|C\|_1 = \max_j \sum_i |C_{ij}|$
   - $\|C\|_\infty = \max_i \sum_j |C_{ij}|$
   - $\|C\|_2 = \max_i \sqrt{\mu_i}$
9. on teste les différentes normes pour avoir $\|C\| < 1$

A quelles conditions sur A, la méthode itérative converge ?

Le cas des matrices à diagonale dominante

Definition

Une matrice carrée A est a diagonale dominante si

. $$\forall i = 1, n \qquad |A_{ii}| > \sum_{j=1, j\neq i}^{n} |A_{ij}|$$

Theorem

Si A est a diagonale dominante alors les méthodes de Jacobi et de Gauss Seidel convergent

Démonstration :

- Jacobi : $\|C_j\|_\infty = \|D^{-1}(M+N)\|_\infty = \max_i \frac{1}{|A_{ii}|} \sum_{j\neq i} |A_{ij}| < 1$
- Gauss Seidel : on montre que si Gauss Seidel diverge, alors A n'est pas à diagonale dominante :
  $1 < \max_i \frac{1}{|A_{ii}|} \sum_{j=1}^{i-1} |A_{ij}| + \sum_{j=i+1}^{n} |A_{ij}|$. Les détails de la preuve ne sont pas évidents...

Exemple : la matrice est-elle à diagonale dominante ?

$$A_1 = \begin{pmatrix} 3 & -1 & 1 \\ 1 & -2 & 0 \\ 1 & 1 & 4 \end{pmatrix} ; A_2 = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix} ; A_3 = \begin{pmatrix} -9 & 3 & 1 & -1 \\ 1 & 2 & 0 & 0 \\ -1 & 1 & 8 & -5 \\ -2 & 1 & 4 & 6 \end{pmatrix}$$

Remarques :

- toute matrice à diagonale dominante est régulière
- on retrouve ce type de matrice dans les méthodes à éléments finis (*cf* meca/EP)

Le cas des matrices positives

Definition (rappel)

Une matrice carrée A est définie positive si elle est symétrique et si si .
$$\forall x \neq 0 \in \mathbb{R}^n \qquad x^\top A x > 0$$

Theorem

Si A est strictement symétrique définie positive alors la méthode de Gauss Seidel converge et la méthode de la relaxation pour $0 < \omega < 2$

Étude des perturbations (où des erreurs)

$$A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix} ; \; b = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix} \text{ admet comme solution } \mathrm{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$b + \delta_b = \begin{pmatrix} 32,1 \\ 22,9 \\ 33,1 \\ 30,9 \end{pmatrix} \quad \text{admet comme solution } \mathrm{x} + \delta_x = \begin{pmatrix} 9,2 \\ -12,6 \\ 4,5 \\ 1,1 \end{pmatrix}$$

Pour la norme infinie :

$$\frac{\|\delta_b\|_\infty}{\|b\|_\infty} = 0,003 \qquad \frac{\|\delta_x\|_\infty}{\|\mathrm{x}\|_\infty} = 13,6$$

C'est la nature de la matrice A qui est en cause : comment mesurer cet effet ?

Étude des perturbations (où des erreurs)

$$A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix} ; \; b = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix} \text{ admet comme solution } \mathrm{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$b + \delta_b = \begin{pmatrix} 32,1 \\ 22,9 \\ 33,1 \\ 30,9 \end{pmatrix} \quad \text{admet comme solution } \mathrm{x} + \delta_x = \begin{pmatrix} 9,2 \\ -12,6 \\ 4,5 \\ 1,1 \end{pmatrix}$$

Pour la norme infinie :

$$\frac{\|\delta_b\|_\infty}{\|b\|_\infty} = 0,003 \qquad \frac{\|\delta_x\|_\infty}{\|\mathrm{x}\|_\infty} = 13,6$$

C'est la nature de la matrice A qui est en cause :
comment mesurer cet effet ?

Conditionnement d'un système linéaire

$$A\mathrm{x} = \mathrm{b} \qquad A(\mathrm{x} + \delta_x) = \mathrm{b} + \delta_b$$

- $\|\mathrm{b}\| = \|A\mathrm{x}\| \leq \|A\| \, \|\mathrm{x}\| \quad \Rightarrow \dfrac{1}{\|\mathrm{x}\|} \leq \dfrac{\|A\|}{\|\mathrm{b}\|}$
- $\|\delta_x\| = \|A^{-1}\delta_b\| \leq \|A^{-1}\| \, \|\delta_b\| \Rightarrow \|\delta_x\| \leq \|A^{-1}\| \, \|\delta_b\|$

$$\frac{\|\delta_x\|}{\|\mathrm{x}\|} \leq \|A^{-1}\|\|A\| \frac{\|\delta_b\|}{\|b\|}$$

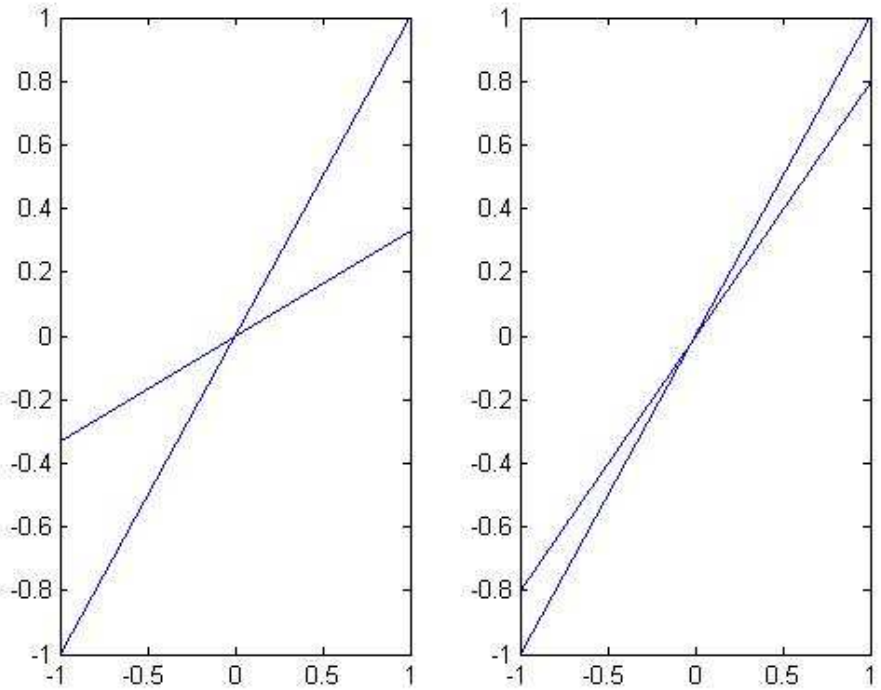
cond(A) : conditionnement de A

$$\mathrm{cond}(A) = \|A^{-1}\|\|A\| = \frac{|\lambda_1|}{|\lambda_n|}$$

- cond(A) proche de 1 : stable
- cond(A) grand devant 1 : instable

Illustration en 2 d

$$Ax = b \qquad \begin{pmatrix} a_{11}x_1 + a_{12}x_2 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2 \end{pmatrix}$$



cond(A) proche de 1 : stable

cond(A) grand devant 1 : instable

Plan

Conclusion

- 3 méthodes itératives (Jacobi, Gauss Seidel, relaxation)
- chaque itération $\leq \mathcal{O}(n^2)$: produit matrice vecteur
  - encore mieux si A est *sparse*
  - peut s'appliquer à des matrices par blocs
- convergent à certaines conditions sur A
- pour comprendre : il faut connaitre la notion de norme matricielle
  - l'importance des valeurs propres
- il existe des méthodes plus sophistiques
  - a chaque itération on se rapproche de la cible...

Différents types de matrices

- quelconque : n lignes p colonnes ..... QR
  - petite, moyenne, grande
  - pleine, creuse, par blocks
  - complexe, réelle, binaire
- carrée : n lignes n colonnes
  - quelconque (carrée) ..... LU
  - symétrique ..... LDL'
  - symétriques positive .... Cholesky : LL', GS
  - diagonale dominante ..... Jacobi
  - triangulaire ..... tri_sup et tri_inf
  - diagonale

Différents types de matrices

- quelconque : *n* lignes *p* colonnes ..... QR
  - petite, moyenne, grande
  - pleine, creuse, par blocks
  - complexe, réelle, binaire
- carrée : *n* lignes *n* colonnes
  - quelconque (carrée) ..... LU
  - symétrique ..... LDL'
  - symétriques positive .... Cholesky : LL', GS
  - diagonale dominante ..... Jacobi
  - triangulaire ..... tri_sup et tri_inf
  - diagonale